REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE **4/2/02**	3. REPORT TYPE AND DATES COVERED **Reprint**
4. TITLE AND SUBTITLE **Block-Based Multi-Period Refresh For Energy Efficient Dynamic Memory**		5. FUNDING NUMBERS **DAAD19-99-1-0304**
6. AUTHOR(S) **Joohee Kim and Marios Papaefthymiou**		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **The University of Michigan Ann Arbor, MI 48109**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211		10. SPONSORING / MONITORING AGENCY REPORT NUMBER **P-39863-EL**.15

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

DRAMs are widely used in portable applications due to their high storage density. In standby mode their main source of power dissipation is the refresh operation that periodically restores leaking charge in each cell to its correct level. Conventional DRAMs use a single refresh period determined by the cell with the largest leakage. This approach is simple but dissipative, because it forces unnecessary refreshes for the majority of the cells with small leakage.

In this paper we investigate a novel scheme that relies on multiple refresh periods and small refresh blocks to reduce DRAM dissipation by decreasing the number of cells refreshed too often. Long periods are used to accommodate cells with small leakage. In contrast to conventional row-based refresh, small refresh blocks are used to increase worst case data retention times. Retention times are further extended by adding a swap cell to each refresh block.

We give a novel polynomial-time algorithm for computing an optimal set of refresh periods for block-based multiperiod refresh. Specifically, given an integer *K* and a distribution of data retention times, in $O(KN^2)$ steps our algorithm computes *K* refresh periods that minimize DRAM dissipation, where *N* is the number of refresh blocks in the memory. We describe and evaluate a possible implementation of our refresh scheme. In simulations with a 16Mb DRAM, block-based multi-rate refresh reduces standby dissipation by a multiplicative factor of 4 with area overhead below 6%.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

20030605 037

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

ARO # P39863-EL

BLOCK-BASED MULTI-PERIOD REFRESH FOR ENERGY EFFICIENT DYNAMIC MEMORY

Joohee Kim and Marios C. Papaefthymiou

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109
{jooheek, marios}@eecs.umich.edu

ABSTRACT

DRAMs are widely used in portable applications due to their high storage density. In standby mode, their main source of power dissipation is the refresh operation that periodically restores leaking charge in each cell to its correct level. Conventional DRAMs use a single refresh period determined by the cell with the largest leakage. This approach is simple but dissipative, because it forces unnecessary refreshes for the majority of the cells with small leakage.

In this paper we investigate a novel scheme that relies on multiple refresh periods and small refresh blocks to reduce DRAM dissipation by decreasing the number of cells refreshed too often. Long periods are used to accommodate cells with small leakage. In contrast to conventional row-based refresh, small refresh blocks are used to increase worst-case data retention times. Retention times are further extended by adding a swap cell to each refresh block.

We give a novel polynomial-time algorithm for computing an optimal set of refresh periods for block-based multi-period refresh. Specifically, given an integer K and a distribution of data retention times, in $O(KN^2)$ steps our algorithm computes K refresh periods that minimize DRAM dissipation, where N is the number of refresh blocks in the memory. We describe and evaluate a possible implementation of our refresh scheme. In simulations with a 16Mb DRAM, block-based multi-rate refresh reduces standby dissipation by a multiplicative factor of 4 with area overhead below 6%.

I INTRODUCTION

Main memories are typically built using Dynamic Random Access Memory (DRAM) technology. To maintain their stored charge levels, DRAM cells require periodic refreshing. Thus, overall DRAM dissipation depends on the frequency of the refresh waveform.

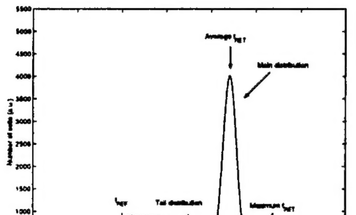


Figure 1: Distribution of data-retention times of DRAM cells.

Conventional DRAMs use a single refresh waveform whose period is dictated by the minimum data-retention time of the memory cells. Figure 1 shows a typical distribution of data retention times in DRAMs. To ensure that the state of all cells is correct at any time, the refresh period t_{REF} must not exceed the shortest data-retention time t_{RET}. For the majority of the cells, however, data retention times are significantly longer than t_{REF}. Refreshing these cells with a period t_{REF} is therefore unnecessary and results in excessive power dissipation.

In this paper we investigate a novel scheme for reducing data-retention power in DRAMs by using multiple refresh periods and refreshing on a block basis. Through the introduction of multiple refresh periods, cells with long data-retention times can be refreshed less often than leaky cells with short ones. Moreover, since the number of leaky cells is typically small [12], by refreshing cells in relatively small groups, as opposed to entire rows, the refresh period of each block is increased. To further extend the refresh periods of refresh blocks, a swap cell is introduced to provide a replacement for the most leaky cell in each block.

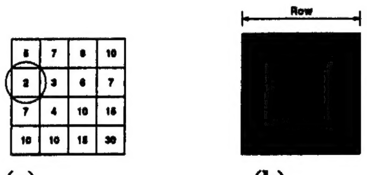
To accompany our scheme, we give a novel polynomial-time algorithm for computing an optimal set of refresh periods that minimizes refresh power dissipation. Specifically, given an integer K and the data retention times of the refresh blocks in the memory, our algorithm determines in $O(KN^2)$ steps a set of K refresh periods that minimize the power dissipation due to refreshing. In addition to this algorithm, we describe a practical implementation of our scheme and evaluate its effectiveness in reducing DRAM power dissipation. In simulations of a 16Mb DRAM, our block-based multi-period refresh reduces power dissipation by a multiplicative factor of 4 with an area overhead of at most 6%.

Numerous approaches have been investigated for reducing data retention power in DRAMs. Schemes that reduce leakage current by optimizing process conditions or by controlling the potential of various nodes in the cell have been reported in [8, 3]. Schemes for dynamically controlling the refresh period through a limited number of temperature or current sensors have been reported in [11, 14]. The use of memory access history for reducing the number of refreshes to previously accessed rows has been proposed in [13]. The use of Error Correcting Codes (ECCs) to control the number of errors below a required level while setting the refresh period to a higher value was reported for mass storage media in [9]. The introduction of a second refresh period for rows with long data retention times was proposed in [6, 2]. These papers focus on implementation issues, however, and do not present systematic techniques for the optimal selection of the second period. A multi-period refresh scheme that relies on ECCs to extend the refresh period was proposed in [7], along with an exponential-time algorithm for the optimal selection of the multiple periods.

The remainder of this paper has six sections. Section II gives an overview of refresh in conventional DRAM. The proposed block-based multi-period refresh scheme is described in Section III. Our polynomial-time algorithm for computing a set of optimal refresh periods is presented in Section IV. Section V describes the architecture of the proposed block-based multi-period memory. Section VI presents simulation results from the application of our methodology to a 16Mb memory. We conclude our paper in Section VII with a brief discussion of future work.

II REFRESH IN CONVENTIONAL DRAM

Typical DRAM cells are composed of one transistor and one capacitor. Charge stored in the cell capacitor degrades over time due to leakage currents. Correct state retention is ensured by periodic recharging (or refreshing) of the stored charge. Among many known leakage currents, the junction leakage current from the storage node is known to be the major leakage mechanism [12]. Due to local process variations, it varies among cells, resulting in fluctuations of t_{RET} among the cells [15, 5]. Studies have showed that the $log_2(t_{RET})$ of the cells follows a bimodal distribution, akin to the one shown in Figure 1. The main distribution, which comprises the majority of the cells, is composed of cells with long t_{RET}. A small tail distribution is composed of leaky cells with short t_{RET} [12].



(a) (b)

Figure 2: Memory with 16 cells. (a) Data retention times and (b) refresh pattern using conventional refresh.

Conventional DRAMs use a single refresh period t_{REF} for every cell. To prevent leakage-induced errors, t_{REF} must be set with respect to the shortest t_{RET} in the memory array. For example, consider the 16-cell memory shown in Figure 2(a), where the integers inside the cells denote their data retention times t_{RET}. In this case, t_{REF} must not exceed 2. Moreover, all cells are refreshed at this rate, as indicated by their uniform shading in Figure 2(b). Consequently, most cells are refreshed too often, resulting in excessive power dissipation.

The data retention power P_{RET} is given by the equation

$$P_{RET} = \frac{A \cdot C}{t_{REF}} + P_{const} \simeq \frac{A \cdot C}{t_{REF}}, \qquad (1)$$

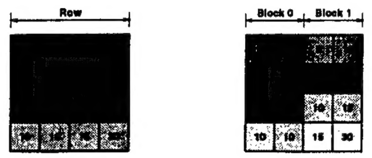
where C is the total switching capacitance, A is a constant proportionality factor, and P_{const} is a power component independent of t_{REF} and typically less than 10% of total dissipation [6].

III BLOCK-BASED MULTI-PERIOD REFRESH

In this section we describe our proposed scheme for reducing data-retention power by introducing additional refresh periods and refreshing on a block basis. In our scheme, groups of adjacent cells, called *refresh blocks*, are refreshed at the same period. The necessary refresh period of each block i, denoted by t_i, is determined by the minimum t_{RET} of the cells in the block. Energy dissipation is reduced by refreshing blocks with long t_i using an accordingly long period ϕ_i such that $\phi_i \le t_i$. Since the data retention times of a few leaky cells are orders of magnitude shorter than those of most cells [12], the necessary refresh period of each block can be extended by decreasing the size of the blocks and by adding a few swap cells to replace leaky cells after fabrication.

III-A Extension of t_i

The effect of the basic block-based multiple-period refresh (BM) scheme on the necessary refresh periods of the example memory from Figure 2 is shown in Figure 3. In Figure 3(a), each row is treated as a single refresh block. In this case, only one row requires a refresh period of 2. The other three rows can be refreshed with longer periods. The magnitudes of the t_i's are encoded in the shading of the refresh blocks, with

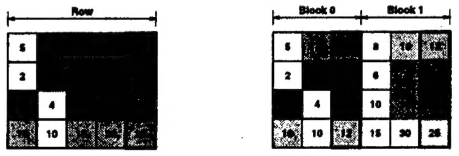


(a) (b)

Figure 3: Extension of necessary refresh periods using BM with (a) 4-cell blocks and (b) 2-cell blocks.

lighter shading indicating longer t_i's. In Figure 3(b), each row is divided into two refresh blocks. As evidenced by the increase in the number of cells with light shading, the number of refresh blocks with longer t_i increases as block size decreases. Only one block requires a refresh period of 2 in this case, with the t_i's of the remaining seven blocks satisfying $t_i \ge 4$.

The necessary refresh periods t_i can be further extended by using one or more swap cells per refresh block. These cells are used to store data which would be otherwise stored in leaky cells in the block. If the t_{RET} of the swap cell is longer than that of the leaky cell, by routing the input and the output of the worst-case cell to these of the swap cell the correct data storage is guaranteed while extending t_i.



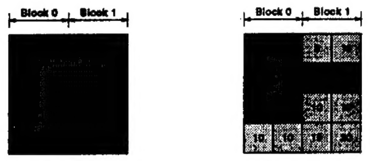
(a) (b)

Figure 4: Extension of necessary refresh periods with eBM using one swap cell and (a) 4-cell blocks or (b) 2-cell blocks.

The impact of the enhanced BM (eBM) scheme, using one additional swap cell per refresh block, is shown in Figure 4. In this figure, cells that are swapped with the additional cell are left unshaded. In comparison with the original memory in Figure 3, eBM increases the t_i's of the blocks, thus increasing the possible ϕ_i's. As block size decreases, t_i's increase further, at the cost of additional area overhead due to the added swap cells. When each row is divided into two blocks, one block requires a refresh period of 2, while for the remaining seven blocks, the necessary refresh period is 5 or longer. If the t_{RET} of the swap cell is the shortest one in the block, then swapping reduces t_i of the block. Since the probability of leaky cell being the swap cell is very small, however, its impact on overall t_i extension is negligible.

III-B Dependency of power on ϕ_i selection

If the number of refresh periods is unlimited, then for each block i, ϕ_i can be set equal to t_i. Since the number of refresh periods is physically limited, however, the ϕ_i of each blocks must satisfy the inequality $\phi_i \le t_i$.



(a) (b)

Figure 5: Assignment of refresh periods from (a) the set $\{2, 5\}$ and (b) the set $\{2, 8\}$.

Figure 5 shows the assignment of two refresh periods with each memory row divided into two refresh blocks. When refresh periods are selected from the set $\{2, 5\}$, the two blocks with t_i equal to 2 and 4 must be refreshed with period 2, while the remaining six with period 5. When refresh periods are selected from the set $\{2,8\}$, the four blocks with t_i equal to 2, 4, 5, and 6 must be refreshed with period 2, and the other four blocks can be refreshed with period 8.

Total data retention power depends on the selection of refresh periods. To achieve maximum power savings, an optimal set of refresh periods must be computed. Revisiting Eq. 1, the data-retention power P_{RET} of a memory that uses block-based multi-period refresh with each block i comprising b cells refreshed at an assigned refresh period ϕ_i is given by

$$P_{RET} = A \cdot c \cdot b \cdot \sum_{i=1}^{M \cdot R} \frac{1}{\phi_i} , \qquad (2)$$

where M is the number of blocks per row, R is the total number of rows in the array, c is the switching capacitance for refreshing one bit, and A is a constant proportionality factor. Using this equation, it follows that for the memory in Figure 5, the refresh period set $\{2, 5\}$ results in lower data-retention power.

IV ALGORITHM FOR SELECTING OPTIMAL PERIODS

In this section we describe a polynomial-time algorithm that computes a set of optimal refresh periods for block-based multi-period refresh, given the size of the set and the data retention times of the refresh blocks in memory.

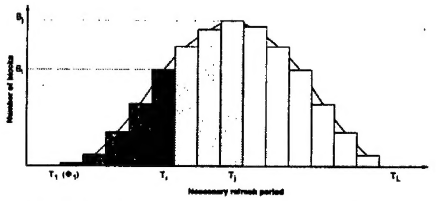


Figure 6: Distribution of necessary refresh times.

The distribution of the data retention times can be determined during post-fabrication testing of the memory arrays. The data retention characteristics of the cells are first measured and then used in bypassing the rows containing the cells with short t_{RET}. (This approach was also used in [6] to assign rows to one of the two refresh periods.) Subsequently, they are processed further before they are provided as input to our proposed algorithm. Specifically, as shown in Figure 6, the space t_{RET} space is divided into L bins. Each bin i contains B_i blocks that must be refreshed at a period no longer than T_i. As we discuss in Section V, the area overhead of our scheme is kept low if the periods $\Phi_1, \Phi_2, \ldots, \Phi_K$ selected by the algorithm are integral multiples of the basic refresh period. This property of the solution can be ensured by selecting the bins at periods that are integral multiples of the original period. The inputs to the algorithm are the resulting distribution $DIST\_L$ and the total number of refresh periods K.

Since $K < L$, in general, there exist bins j whose blocks will be refreshed at a period shorter than T_j. The power associated with refreshing blocks in bins $i+1$ through j with a period T_{i+1} is the sum of the power dissipation in all blocks of these bins and is given by the expression

$$p_{i+1,j} = \frac{D}{T_{i+1}} \sum_{n=i+1}^{j} B_n , \qquad (3)$$

where D is constant proportionality factor. Given a selection of K refresh periods, the total data-retention power is the sum of these partial powers up to bin L. Among the ${}_LC_K$ possible combinations, we need the compute the one(s) for which total power dissipation is minimized.

The key to obtaining an efficient algorithm for the overall optimization problem is to express optimal solutions to subproblems in a recursive manner. Dynamic programming can then be used to solve the recursion in polynomial steps. Let $P_{j,k}$ be the *minimum* power required to refresh the blocks in the first j bins using k periods. As can be verified with the help of Figure 6, the optimal dissipation $P_{j,k}$ is obtained by introducing an additional refresh period T_{i+1} to the optimal solution obtained with $k-1$ periods up to some bin i. This fact is captured by the following recursion.

$$P_{j,k} = \min_{i<j} \left\{ P_{i,(k-1)} + \frac{D}{T_{i+1}} \sum_{n=i+1}^{j} B_n \right\} . \qquad (4)$$

Since the minimum total power is required, the period T_{i+1} that minimizes $P_{j,k}$ is selected as the kth period.

```
SP_OPT ($K$, $DIST\_L$)
 1: $accm_0 \leftarrow 0$
 2: $S_{0,0} \leftarrow \emptyset$
 3: for $l = 1$ to $L$ do
 4:     $accm_l \leftarrow accm_{l-1} + B_l$
 5:     $P_{l,1} \leftarrow D \cdot accm_l / T_1$
 6:     $S_{l,1} \leftarrow \{T_1\}$
 7: end for
 8: for $k = 2$ to $K$ do
 9:     $P_{0,k-1} \leftarrow 0$
10:     $S_{0,k-1} \leftarrow S_{0,k-2} \cup \{T_1\}$
11:     for $j = 1$ to $L$ do
12:         $P_{j,k} \leftarrow \infty$
13:         for $i = 0$ to $j-1$ do
14:             $partP \leftarrow D \cdot (accm_j - accm_i)/T_{i+1}$
15:             if $P_{j,k} > P_{i,k-1} + partP$ then
16:                 $P_{j,k} \leftarrow P_{i,k-1} + partP$
17:                 $S_{j,k} \leftarrow S_{i,k-1} \cup \{T_{i+1}\}$
18:             end if
19:         end for
20:     end for
21: end for
22: return $S_{L,K}$
```

Figure 7: Algorithm for computing optimal set of periods.

Figure 7 gives pseudocode for solving the recursion in Equation 4. The variable $S_{j,k}$ maintains the optimal set of k periods minimizing $P_{j,k}$. When the algorithm terminates, the K optimal periods are found in $S_{L,K}$. The compute intensive part of the algorithm is the three nested loops, and its total runtime is $O(KL^2) = O(KN^2)$, since $L \leq N$.

V ARCHITECTURAL ORGANIZATION

In this section we describe a hardware organization that implements block-based multi-period refresh. Our architecture uses clock periods that are integral multiples of the shortest period, allowing the blocks with longer assigned period to skip refresh until their period arrives. Such skipping was also used in [6] to implement two-period refresh.

Figure 8 gives a schematic description of the proposed architecture. The refresh period signals are generated in the refresh period generator. For each block, the assigned refresh

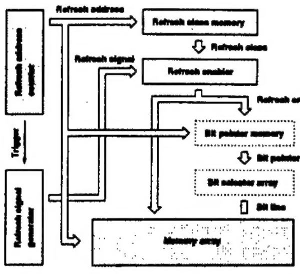


Figure 8: Schematic description of BM DRAM architecture. Shaded modules constitute conventional DRAM.

period is mapped in the refresh class memory. The stored period is compared with the generated period each time the row containing the block is addressed for refresh. If they agree, block refresh is enabled by the refresh enabler. The dotted blocks need to be added to support the swap cell scheme. When refreshing of a block is enabled, the pointer to the cell to be swapped is read from the bit pointer memory, and the data are swapped by the bit selector. In the following subsections, we give a more detailed description of our architecture for memories configured with R rows, M blocks per row, b cells per block, one swap cell per block, and K refresh periods.

V-A Implementation of BM

Due to process variations, every memory will have different leakage current distribution. Hence, to guarantee maximum power reduction for every memory, the refresh period generator must be programmable.

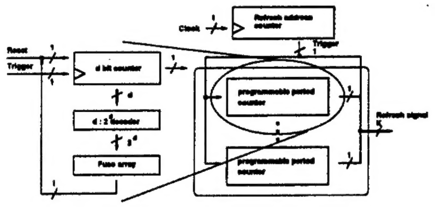


Figure 9: Refresh signal generator.

Figure 9 describes a refresh period generator. It is composed of $K-1$ counters of programmable period to generate $K-1$ additional refresh periods, added to the original period generated from the refresh address counter. The counters are incremented each time the refresh address counter completes a cycle, thus generating integral multiples of the original period. The cycle period of each counter can be programmed to one of the added periods by programming its reset condition. This is accomplished by connecting an output of the $d : d^2$ decoder, which is asserted once every period, to the counter reset by means of fuses. The output of the refresh period generator is a K-bit signal indicating whether the current cycle is the programmed period or not for the counters.

Information about the refresh period assigned to each block is stored in the refresh class memory. To minimize memory size, the actual information stored in the refresh memory is not the period itself but the pointer to the counter in the refresh signal generator that generates the period. Thus, its total size is $R \times M \times log_2K$. The refresh class memory is indexed by the refresh address and outputs a pointer of log_2K bits for each block.

Block refreshing, which should be performed only when the current period matches the assigned period, is enabled by the refresh enabler. As shown in Figure 10, the refresh enabler is a simple array of $K : 1$ multiplexors. The input to the

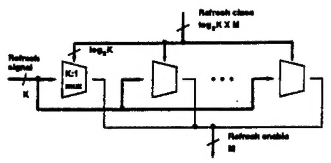


Figure 10: Refresh enabler

multiplexors is a K-bit refresh signal from the refresh signal generator, and its control signal is log_2K-bit refresh class from the refresh class memory. Thus, the refresh of a block is enabled only when its corresponding refresh signal is asserted.

The word lines and sense-enable lines in the memory array are divided into sub-lines covering each block. The refresh of the blocks is enabled only when both the row containing the block is addressed for refresh and the refresh of the block is enabled. A similar approach of dividing the word lines is proposed in [10] to reduce the switching capacitance in memory.

V-B Implementation of eBM

The position of the cell to be swapped for each block is stored in the bit pointer memory. To reduce power consumption, the bit pointer memory is read only when the refresh of the block is enabled. This is accomplished by dividing the word lines to blocks and activating them only when the refresh of the block is enabled. The bit pointer memory is indexed by the refresh address and outputs a log_2b-bits pointer. Thus, its total size is $R \times M \times log_2b$.

The refresh on the bad cell is redirected to the swap cell by the bit selector. The bit selector of a block consists of a log_2b:b decoder and b bidirectional multiplexors.

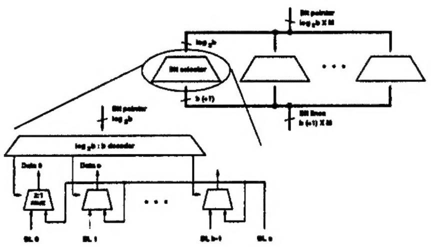


Figure 11: Bit selector array

As shown in Figure 11, the decoder takes the bit pointer from the bit pointer memory as input and selects the multiplexor of the cell to be swapped, thus redirecting the access on the bit line of the bad cell to that of the swap cell.

The extra memory cells required for employing eBM with one swap cell per block is $R \times M \times 1$.

VI SIMULATION RESULTS

We have evaluated the effectiveness of block-based multi-period refresh on a 16Mb DRAM fabricated in $0.6\mu m$ technology whose distribution of data retention times is given in [12]. The data-retention power of this memory in the self-refresh mode is $5.5mW$ at a refresh period of $64ms$ [1]. For our block-based multi-period refresh scheme the power dissipation and the area overhead of the added logic modules was computed using the Epoch CAD tool at a $0.7\mu m$ standard cell technology. The power dissipation of the added memory modules was estimated by scaling the power dissipation of the 16Mb memory. Our simulation results show that block-based multi-period refresh can result in significant savings in data-retention power with minimal area overhead. In this section we present our results and describe the configurations achieving minimum power.

VI-A Data-retention power

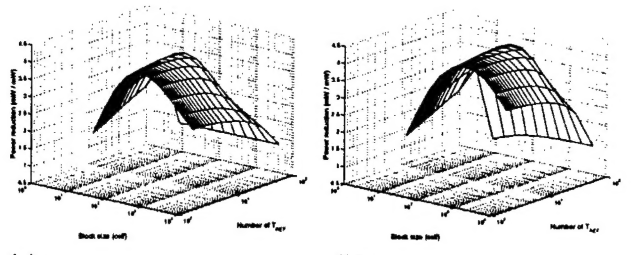

(a) (b)

Figure 12: P_{RET} reduction ($P_{conv}/P_{BM|eBM}$) versus block size and number of allowed refresh period for (a) BM and (b) eBM

Figure 12 shows the relative power reductions of BM and eBM over conventional single-period refresh. Data retention power decreases by up to a factor of 4. Our simulations took into account both the frequency dependent and the frequency independent component of memory power dissipation. With a fixed number of refresh periods, block size has a convex effect on power reduction, resulting in maximum power reduction in the range of about one hundred cells per block. For smaller block sizes, the power overhead of the additional memories overtakes the benefits of longer retention times. With a fixed block size, power reduction reaches an asymptotic maximum as the number of refresh periods increases. In general, eBM has similar behavior with BM and achieves larger maximum power reduction at larger block sizes.

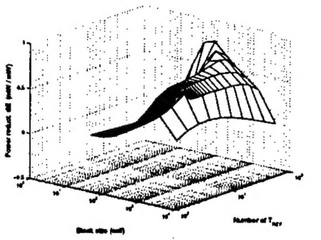


Figure 13: Difference in relative power reduction between BM and eBM (P_{conv}/P_{BM} - P_{conv}/P_{eBM}).

Figure 13 shows the difference in the reduction of data-retention power using BM and eBM. The introduction of swap cell is beneficial for larger block sizes, due to the longer ϕ_i's. eBM consumes more power when block sizes are small, since the reduction in the data retention power is offset by the power consumed in the large bit pointer memory.

VI-B Area overhead

In this section we discuss the area overhead of our proposed scheme, using transistor counts as the measure.

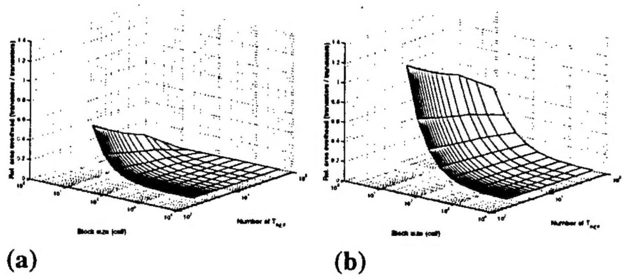

(a) (b)

Figure 14: Relative area overhead (extra tr. / 16M tr.) versus block size and number of allowed refresh period for (a) BM and (b) eBM

The area overhead of our scheme is shown in Figure 14. Area overhead decreases as the block size increases. It increases as the number of refresh periods increases. The area overhead of eBM increases drastically as the block size decreases due to the bit pointer memory. It still stays below 10% for large block sizes. Though this work was done on $16Mb$ DRAM, the area overhead will scale linearly with the memory size as discussed in Sec. V.

VI-C Maximum power reduction configuration

The memory configurations that achieve maximum reduction in data-retention power are shown in Figure 15.

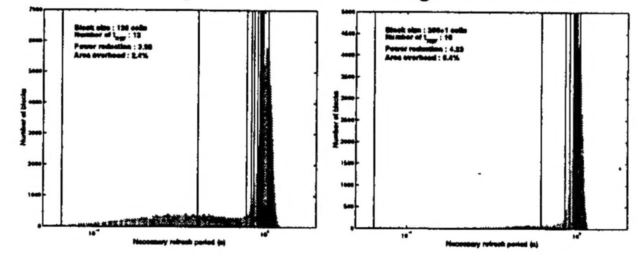

(a) (b)

Figure 15: Optimum configuration achieving minimum power for (a) BM and (b) eBM

The vertical lines denote the selected refresh periods. As expected, the optimum block size for eBM is larger than that of BM, due to the swapping of leaky cells which reduces blocks with short ϕ_i. With BM, P_{RET} is reduced by a factor of 3.93 using blocks of 128 cells and 12 refresh periods. The area overhead, in transistor count, is 2.4%. For eBM, P_{RET} is reduced by a factor of 4.23 using blocks of 256 cells with 1 swap cell per block and 10 refresh periods. The area overhead is 5.4%. Except for a few short refresh periods, the rest of the periods are closely placed due to the converging ϕ_i's.

VII CONCLUSION

This paper describes a novel block-based multi-period refresh scheme for low data-retention power in dynamic memories. A novel polynomial-time algorithm is given for selecting an optimal set of refresh periods that minimizes data retention power. In addition, an implementation of the proposed scheme is described. Simulation results with a 16Mb DRAM show power reductions up to a factor of 4 over conventional refresh, with an area overhead less than 6%. We are currently evaluating the effect of process variations on the effectiveness of our scheme.

Acknowledgments

This research was supported in part by the National Science Foundation under Grant No. MIP-9610108 and by the US Army Research Office under Grant No. DAAD19-99-1-0304.

REFERENCES

[1] *16,777,216-word X 1-bit DYNAMIC RAM Data sheet.* Toshiba.

[2] S. Takase and N. Kushiyama. A 1.6Gb/s DRAM with flexible mapping redundancy technique and additional refresh scheme. In *International Solid-State Circuits Conference*, pages 410–411, 1999.

[3] H. Yamauchi et al. A circuit technology for a self-refresh 16Mb DRAM with less than 0.5 μ A/Mb data retention current. *IEEE Journal of Solid-State Circuits*, 30(11):1174–1182, November 1995.

[4] A.Y. Romanenko and W.M. Gosney. A numerical analysis of the storage times of dynamic random-access memory cells incorporating ultra-thin dielectrics. *IEEE Transactions on Electron Devices*, 45(1):218–223, January 1998.

[5] P.J. Restle, J.W. Park, and B.F. Lloyd. DRAM variable retention time. In *International Electron Devices Meeting 1992*, pages 807–810. 1992.

[6] Y. Idei et al. Dual-period self-refresh scheme for low-power DRAM's with on-chip PROM mode register. *IEEE Journal of Solid-State Circuits*, 33(2):253–259, February 1998.

[7] J. Kim and M.C. Papaefthymiou. Dynamic memory design for low data-retention power. In *PATMOS 2000*, pages 207–216, September 2000.

[8] L.C. Hsia et al. Effects of hydrogen annealing on data retention time for high density DRAMs. In *1997 International Symposium on VLSI Technology, Systems, and Applications. Proceedings of Technical Papers*, pages 142–147. 1997.

[9] Y. Katayama et al. Fault-tolerant refresh power reduction of DRAMs for quasi-nonvolatile data retention. In *International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 311–318, 1999.

[10] T. Murotani et al. Hierchical word-line architecture for large capacity DRAMs. *IEICE Trans. Electron.*, E80-C(4):550–556, 1997.

[11] Y. Kagenish et al. Low power self refresh mode DRAM with temperature detecting circuit. In *1993 Symposium on VLSI Circuits. Digest of Technical Papers*, pages 43–44. 1993.

[12] T. Hamamoto, S. Sugiura, and S. Sawada. On the retention time distribution of dynamic random access memory (DRAM). *IEEE Transactions on Electron Devices*, 45(6):1300–1309, June 1998.

[13] T. Ohsawa, K. Kai, and K. Murakami. Optimizing the DRAM refresh count for merged DRAM/logic LSIs. In *Proceedings 1998 ISLPED*, pages 82–87. 1998.

[14] J. Nyathi and J.G. Delgado-frias. Self-timed refreshing approach for dynamic memories. In *Proceedings Annual IEEE international ASIC Conference*, pages 169–173. 1998.

[15] E. Adler et al. The evolution of IBM CMOS DRAM technology. *IBM J. Develop.*, 39(1/2):167–188, March 1995.

[16] H. Ozaki et al. The optimization of a DRAM CMOS row decoder circuit. *Electronics and Communications in Japan, Part2*, 74(3):1–9, 1990.